1 FW AF 91

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
Before the Board of Patent Appeals and Interferences

In re Patent Application of: SWAINE
Atty Dkt. 550-235
C# M#
TC/A.U.: 2183
Serial No. 09/876,220
Examiner: Kevin P. Rizzuto
Filed: June 8, 2001
Date: January 4, 2006

Title: APPARATUS AND METHOD FOR EFFICIENTLY INCORPORATING INSTRUCTION SET INFORMATION WITH INSTRUCTION ADDRESSES

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

☐ **Correspondence Address Indication Form Attached.**

☐ **NOTICE OF APPEAL**
Applicant hereby **appeals** to the Board of Patent Appeals and Interferences from the last decision of the Examiner twice/finally rejecting applicant's claim(s). $500.00 (1401)/$250.00 (2401) $

☒ An appeal **BRIEF** is attached in the pending appeal of the above-identified application $500.00 (1402)/$250.00 (2402) $ 500.00

☐ Credit for fees paid in prior appeal without decision on merits -$ ( )

☐ A reply brief is attached. **(no fee)**

☐ Petition is hereby made to extend the current due date so as to cover the filing date of this paper and attachment(s)
One Month Extension $120.00 (1251)/$60.00 (2251)
Two Month Extensions $450.00 (1252)/$225.00 (2252)
Three Month Extensions $1020.00 (1253/$510.00 (2253)
Four Month Extensions $1590.00 (1254/$795.00 (2254) $

☐ "Small entity" statement attached.

Less month extension previously paid on -$( )

TOTAL FEE ENCLOSED $ 500.00

Any future submission requiring an extension of time is hereby stated to include a petition for such time extension. The Commissioner is hereby authorized to charge any deficiency, or credit any overpayment, in the fee(s) filed, or asserted to be filed, or which should have been filed herewith (or with any paper hereafter filed in this application by this firm) to our **Account No. 14-1140.** A duplicate copy of this sheet is attached.

901 North Glebe Road, 11th Floor
Arlington, Virginia 22203-1808
Telephone: (703) 816-4000
Facsimile: (703) 816-4100
JRL:sd

NIXON & VANDERHYE P.C.
By Atty: John R. Lastova, Reg. No. 33,149

Signature: John R. Lastova

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of

SWAINE

Atty. Ref.: 550-235

Serial No. 09/876,220

Group: 2183

Filed: June 8, 2001

Examiner: Kevin P. Rizzuto

For: APPARATUS AND METHOD FOR EFFICIENTLY INCORPORATING INSTRUCTION SET INFORMATION WITH INSTRUCTION ADDRESSES

Before the Board of Patent Appeals and Interferences

BRIEF FOR APPELLANT

On Appeal From Final Rejection
From Group Art Unit 2183

John R. Lastova
NIXON & VANDERHYE P.C.
11th Floor, 901 North Glebe Road
Arlington, Virginia 22203-1808
(703) 816-4025
Attorney for Appellants
Swaine
ARM Limited

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re Patent Application of

SWAINE — Atty. Ref.: 550-235

Serial No. 09/876,220 — Group: 2183

Filed: June 8, 2001 — Examiner: Kevin P. Rizzuto

For: APPARATUS AND METHOD FOR EFFICIENTLY INCORPORATING INSTRUCTION SET INFORMATION WITH INSTRUCTION ADDRESSES

* *

January 4, 2006

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPEAL BRIEF

I. REAL PARTY IN INTEREST

The real party in interest is the assignee, ARM Limited, a United Kingdom corporation.

II. RELATED APPEALS AND INTERFERENCES

There are no other appeals related to this subject application. There are no interferences related to this subject application.

III. STATUS OF CLAIMS

Claims 1-22 are pending. Claim 21 stands rejected under 35 U.S.C. §112, second paragraph. Claims 1-22 stand rejected under 35 U.S.C. §103 as being unpatentable over an ARM specification identified as "Embedded Trace Macrocell," referred to by the Examiner simply as ARM.

IV. STATUS OF AMENDMENTS

No amendment has been filed after final.

V. SUMMARY OF THE CLAIMED SUBJECT MATTER

The claims are directed to data processing systems that are able to execute processing instructions from a number of different instruction sets. There are many situations where it is desirable to keep track of the processing performed by the processing circuit, and in such situations, it may be desirable to be able to identify at any point in time which instruction set is being used. For example, such information is useful during the development of data processing systems, where it is often desirable to track the activity of the processing circuit. An example of a tool that may be used to assist in such a process is a tracing tool. See, e.g., page 1, lines 7-14.

Tracing a data processing system produces a trace stream of data representing the step-by-step activity within the system. Increased amounts of tracing functionality are being placed on-chip, e.g., the Embedded Trace Macrocell provided by ARM Limited and applied by the Examiner in the obviousness rejection. A real time trace stream of

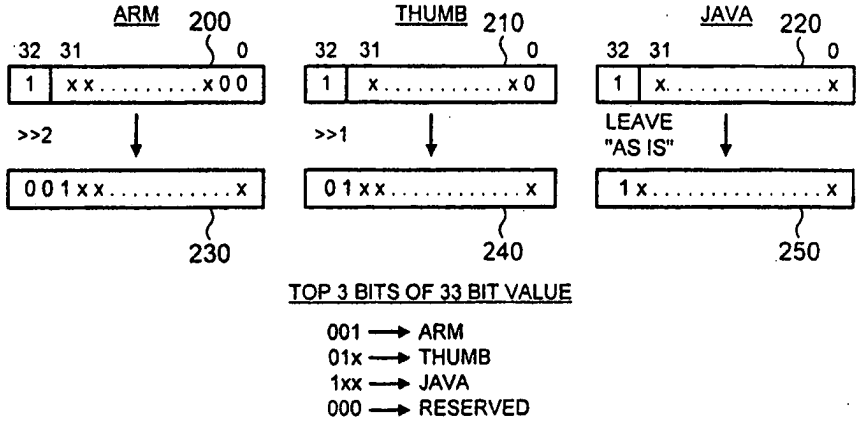
certain data representing activities facilitates program debugging. If the processing circuit can execute instructions from different instruction sets, then it would be desirable for tracing mechanisms to also keep track of that instruction set information. See, e.g., page 1, lines 15-27 and page 2, lines 10-16.

Independent claim 1 recites a multiple instruction set processing circuit. The non-limiting example embodiment shows a data processing system 2 with an integrated circuit 4 that includes a microprocessor core 6, a cache memory 8, an on-chip trace module controller 10, and an on-chip trace buffer 12. The integrated circuit 4 is connected to an external memory 14 which is accessed when a cache miss occurs within the cache memory 8. A general purpose computer 16 is coupled to the on-chip trace module controller 10 and the on-chip trace buffer 12 and serves to recover and analyze a stream of tracing data from these elements using software executing upon the general purpose computer 16. Page 12, lines 8-16.

The claims also recite encoding logic. Figure 2 shows a function block diagram of the on-chip trace module 10 having a sync block 100, a trigger block 110, and a control block 120. In response to trigger signals, the control logic 120 outputs necessary trace data to the trace buffer over path 145. The control logic 120 includes additional logic to encode instruction set information with instruction addresses and to then compress such encoded instruction addresses prior to output over path 145. Page 12, line 26-page 13, line 24.

The encoding performed within the control logic 120 is illustrated schematically with reference to the non-limiting example in Figure 3, reproduced below, where three

different instruction sets that may be used by the processor core 6 are identified by the names "ARM", "Thumb" and "Java".



Java instructions 32 bits in length may begin at any address location in memory, and all 32 bits need to be specified in a Java instruction address 220. However, Thumb instructions may only begin at even address locations in memory, for example location 0, location 2, location 4, location 6, etc. Accordingly, bit position 0 in a 32 bit Thumb instruction address 210 is always zero. Similarly, ARM instructions may only begin at every fourth address location in memory, for example, location 0, location 4, location 8, location C, etc. Accordingly, the least two significant bits of an ARM instruction address 200 are always zeros. Page 15, line 25-page 14, line 7.

A logic one value is pre-pended—added to the beginning of the instruction at the most significant bit—as a thirty-third bit of each instruction address (i.e. is placed at bit position 32). Then, encoded instruction addresses are produced by applying an appropriate right shift to remove any irrelevant least significant bit(s) of the instruction address. For Java instructions, since all 32 bits need to be specified, no right shifting is performed, and the encoded instruction address 250 is merely the 33 bit value given by

pre-pending the logic one value to the original 32 bit Java instruction address 220. For Thumb instructions, a right shift of one is applied to discard the least significant bit, which produces a 33-bit encoded instruction address 240 where the most significant two bits are "01". For ARM instructions, a right shift of two bit positions is applied to discard the two least significant bits, whereby a 33-bit encoded instruction address 230 is produced with the most significant three bits being "001". Page 14, lines 8-25.

Hence, the top three bits of the 33-bit encoded instruction address indicate which instruction set the instruction address relates: "001" in the top three bits identifies an ARM instruction, "01x" identifies a Thumb instruction (x signifying any value), and "1xx" identifies a Java instruction.

Many of the dependent claims recite compression logic, an example of which is shown in Figure 6. The compression of the 33-bit encoded instruction address prior to being output to the trace buffer over path 145 is part of a preferred but not required implementation. The compression technique is illustrated in Figure 4 reproduced here for convenience.

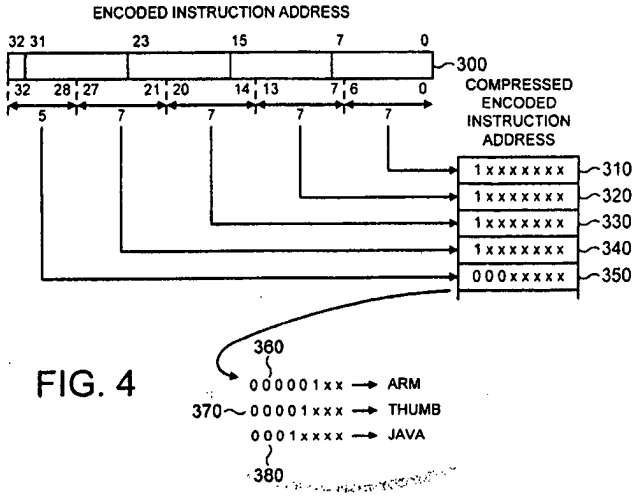


FIG. 4

The 33-bit encoded instruction address 300 (which may be any of the encoded instruction addresses 230, 240, 250 illustrated in Figure 3) is split into 7-bit sections, the most significant 5 bits being extended to a 7-bit section by pre-pending two logic zero values to the 5 bits. Each 7-bit section to be output as the compressed encoded instruction address is extended to a byte (i.e. 8 bits) by pre-pending a flag to indicate whether the corresponding 7-bit section is the last 7-bit section being output as the compressed encoded instruction address. In preferred embodiments, the flag takes the form of a continuation bit, which is set to a logic one value to indicate that a further section is to be output, and is reset to a logic zero value to indicate that the corresponding section is the last section of the compressed encoded instruction address. The bits in section 350 identify the instruction set. Sequence 360 identifies the ARM instruction set, sequence 370 the Thumb instruction set, and sequence 380 the Java instruction set. Page 14, line 29-page 16, line 4.

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

The indefiniteness rejection of claim 21 and the obviousness rejection of all claims 1-22 are requested for review on appeal.[1]

[1] Appellant does not believe that an objection to the title of the application is appealable. While the Examiner's have the authority to change the title themselves, Appellant believes that the title suggested by the Examiner is somewhat misleading because the main claims are not restricted to trace data.

VII. ARGUMENT

A. Claim 21 Would Be Readily Understood By a Person of Ordinary Skill in the Art

The Examiner's objection is that the Examiner cannot decide whether claim 21 is an independent claim or a dependent claim. But that concern, even if it were a valid concern, does not mean that the metes and bounds of the claim are unclear or indefinite. The type of thing being claimed, a computer readable medium carrying a computer program, is well known by those skilled in the art. The computer program is clearly defined as controlling an apparatus in accordance with the method of claim 17. The Examiner raises no issues regarding the clarity of method claim 17. This short hand reference to the method in claim 17 simply means that all of the steps recited in claim 17 need not be repeated in claim 17. Claim 17 is clear and distinct. The rejection should be reversed.

B. The ARM Reference

The ARM reference describes an Embedded Trace Macrocell (ETM) design developed by the assignee of the present application. The Examiner relies primarily on section 2.5.1 on page 2-8 of the ARM citation which describes that the ETM design supports both ARM and Thumb instructions. When handling branch instructions, bit position zero of the address is used to show whether the destination of the branch is ARM

code (bit zero low) or Thumb code (bit zero high). This bit position zero encoding is possible because, for both ARM and Thumb instructions, bit zero of the address is always a logic zero value, and hence, can be inferred without having to check the value at bit position zero. Indeed, this property of ARM and Thumb instructions is described in the present application at page 13, line 27 to page 14, line 4. All remaining address bits (i.e., bit positions 1-31) are unaltered by this technique, and the encoded output in both instances still equals 32 bits.

One problem with the technique disclosed in the ARM citation is that it is only applicable to instruction sets where bit zero of the address will always be zero. But this technique does not have general applicability. For example, it could not be used for the Java instruction set where it is not always the case that bit zero of the address will have a logic zero value. Another problem is that the output encoded address in the ARM citation is always the same size as the original address, (i.e., 32 bits in the example given), and all of the bits in the output encoded address are required, which does not represent a particularly efficient encoding. It would be beneficial if a more efficient encoding could be realized where in some instances the encoded address as output had a certain number of bits which could be ignored, since this would allow efficient further processing, for example compression.

C. The ARM Citation Fails to Disclose or Suggest Multiple Features in the Independent Claims

Claim 1 recites both that the instruction address has "a predetermined number of bits, irrespective of the instruction set to which the associated processing instruction

belongs," and that "a different number of *most significant* instruction address bits needs to be specified in the instruction address to uniquely identify processing instructions in different instruction sets." The instruction address encoding is achieved by performing a computation equivalent to "***removing** any least significant bits* not forming the instruction address bits needing to be specified, and ***extending** the specified instruction address bits to n-bits by **prepending** a pattern of bits* to the specified instructions address bits, the number of least significant bits removed and the pattern of bits prepended being dependent on the instruction set corresponding to that instruction." Similar recitations are recited in the other two independent claims 15 and 17.

Removing any least significant bits that can be removed while still enabling the instruction to be uniquely identified, and then prepending a pattern of bits to the remaining specified instruction address bits (i.e., adding the pattern of bits to the opposite end of the address than that from which any appropriate least significant bits were removed), *shifts* the bit positions of the specified instruction address bits by some predetermined amount. None of these features is disclosed or suggested in the ARM citation.

D. The ARM Citation Does Not Describe the Claimed Computation Equivalent

The ARM citation does not teach the claimed computation equivalent including the removing and the extending-by-prepending operations. The Examiner argues that the replacement of the least significant bit of an ARM or Thumb instruction address in the

ARM citation by an indication of the type of instruction that address corresponds to the claimed computation equivalent. But this is not the case.

As explained above, the ARM citation takes advantage of the fact that for both an ARM instruction and a Thumb instruction, bit zero of the address is always a logic zero value the bit zero position can be encoded. All other address bits (i.e., bits 1 to 31) remain unaltered. A bit indicating the instruction set "replaces" the bit zero value. Given that all of the other bits of the address are unaffected, and given that bit zero is used to indicate the instruction set, it is a strained reading to stay that bit zero is "removed." But even with this strained reading, the technique in the ARM citation does not describe the "extending" and "prepending" that go along with the "removing." Indeed, the Examiner fails to acknowledge that the ARM citation does not teach the claimed extending.

The Examiner does admit that the ARM citation fails to teach the claimed prepending a pattern of bits to the specified instruction address bits. The effect of the computation equivalent to removing, extending, and prepending is to shift the bit positions of the specified instruction address bits by some predetermined amount. In contrast, the bit positions in the ARM citation are static—they don't shift.

E. The Missing Claim Features Are Not Obvious

1. The Arm Citation Does Not Prepend Bits

It would not have been obvious to modify the ARM citation to include the missing claim features. The on-line Computer Desktop Encyclopedia defines "append" as "to add to the end of an existing structure" and "prepend" as the opposite of append: "to attach to the beginning of data." The least significant bit in the ARM citation is

replaced with an instruction set indicator bit. An additional instruction set indicator bit is not attached to the beginning of the instruction. Nor is an additional instruction set indicator bit added to the end of the existing instruction. Certainly, the ARM citation does not teach adding a pattern of bits to the opposite end of the instruction address from which least significant bits may have been removed.

2. The *Japikse* Case Is Inapposite

The Examiner argues that it would have been obvious to prepend bits instead of append bits, relying on *In re Japikse*, 181 F.2d 1019 (CCPA 1950). This case does not save the Examiner's argument. First, as pointed out above, the ARM citation teaches replacing a bit value in bit position zero, which is not the same as adding a bit or appending a bit. Second, because *In re Japikse* is a 1950 case that predates the 1952 patent act and the seminal *Graham v. John Deere* Supreme Court case, both of which greatly restructured the law on obviousness, the *In re Japikse* case is not particularly relevant. A large body of obviousness case law has been built over the last 50 years upon legal foundations that were not in place at the time *In re Japikse* was decided.

Third, the analysis relied on by the Examiner from the *In re Japikse* decision does not apply to the facts in this case. The *Japikse* court found no error in the finding that "there would be no invention in shifting the starting switch disclosed by Cannon to a different position since the operation of the device would not thereby be modified." *Id.* at 1023. That finding does not apply to the facts here. The encoded instruction address produced in accordance with the claimed technique is of no use as an instruction address to the processing circuit because the individual bit positions of the address are shifted.

Why would one of ordinary skill in the art be motivated to modify the instruction address in the ARM citation, in which the instruction address is a functional and useful address to the processing circuitry, into something that no longer can be read by the processing circuitry as an instruction address? Nevertheless, the inventor realized that encoding instruction addresses in the manner defined in Claim 1 can be useful in a variety of situations, e.g., when tracing the activity of a data processing apparatus, since the output encoded instruction address can still be analyzed by trace analysis logic even though the encoded instruction address itself would not be useable by the processing circuitry being traced. The modification that the Examiner proposes for the ARM citation renders the instruction address inoperable for its intended purpose as an instruction address. The Federal Circuit has found on several occasions that a proposed modification that renders the prior art reference inoperable for an intended purposed is inappropriate for an obviousness inquiry. See, e.g., *In re Fritch*, 972, F.2d 1260, 165-66 (Fed. Cir. 1992).

Nor does the fact that bits *can* be prepended to an instruction address in the ARM citation make it obvious to actually append them. The prior art as a whole must suggest the desirability of the combination. *In re Beattie*, 974, F.2d 1309, 1311 (Fed. Cir 1992). Feasibility is not the same as desirability. *Winner Int'l Royalty Corp. v. Wang*, 202 F.3d 1340, 1349 (Fed. Cir. 2000).

3. The Official Notice Position Is Also Inapposite

The Examiner offers up an Official Notice position that it is well known to include a header to information that is about to be transferred at the beginning of the data. See paragraph 19 in the final action. The Examiner then extrapolates that it would have

been obvious to prepend bits rather than append bits in the ARM citation, since placing "information about data about to be transferred at the beginning of data to be transferred" allows a receiver to "take appropriate action." See paras. 19 and 20. The Official Notice contention does not save the rejection.

First, the Examiner is using official notice as a second reference. Of course, if a second reference actually teaching a header to a data packet were identified and applied, it would likely be a reference related to data communications rather than a reference related to different instruction sets and embedded tracing operations. So the analogy the Examiner is making requires bridging two different technologies. Data communication packets have separate headers and payloads. The headers are added and removed by different communications protocol layers. This same is not true for instruction addresses which are not packets. Moreover, as set out in the claims of the present application, the instruction set indication information is encoded in a way that merges it with the instruction address information. It cannot reasonably be viewed as a separate header.

Furthermore, the Examiner's analogy is flawed. In the claimed data processing system environment, the bits output first depend on the endianness[2] of the system. For example, if the data processing apparatus is little endian, then the least significant bits are output first. Hence, the bits of the encoded address which give an indication of instruction set will actually be output last.[3] This incongruity illustrates why the

[2] The way numbers are stored in a computer word. With big endian, the most significant byte or digits are placed leftmost in the structure (the big end). With little endian, the most significant byte or digits are placed rightmost in the structure (the little end).
[3] For the purposes of the present invention, it does not matter whether the data processing apparatus is little endian or big endian.

Examiner's header and payload analogy simply does not work with the ARM citation's instruction addresses.

4. The ARM Citation Does Not Offer the Advantages of the Claimed Instruction Address Encoding Technique

The inventor realized that the claimed encoding technique yielded surprising and unexpected benefits. Even though an encoded instruction address produced in the manner claimed is not useful as an instruction address to the processing circuit, it does represent a particularly efficient and flexible technique for performing encoding. In particular, it is not subject to the restrictions placed on the encoding technique described in the ARM citation, which is only applicable if bit zero of the address is always a logic zero value. So instruction sets like Java which do not have that bit zero logic value may also be encoded.

Further, the format of the encoded address lends itself to efficient further processing. For example, as defined in claim 4 and discussed with reference to Figure 4, such an encoded address lends itself to a particularly efficient compression. This is due in part to the fact that the encoding can produce an n-bit encoded instruction address where a certain number of the most significant bits can be ignored. In contrast with the technique described in the ARM citation where the encoded address always contains 32 bits of data which needed to be output. In the preferred example embodiment described in the present application, only 31 bits are required if the instruction being encoded is an ARM instruction (since the most significant bit will be 0), which gives a 1-bit saving for each ARM instruction. These advantages achieved by the claimed invention, lacking in the ARM reference, are further evidence of non-obviousness.

F. Dependent Claim Features Are Not Found in the ARM Citation

The rejections of the dependent claims are built on the premise that the ARM citation teaches the claimed encoding by performing a computational equivalent to removing, extending, and prepending. But as demonstrated above, this premise is faulty on multiple grounds. The dependent claims also recite additional features not found in the ARM citation.

Claim 2 for example recites that the claimed first and second patterns of prepended bits are related by shifting the bits. The Examiner admits that this feature is missing. Lacking a teaching of this feature anywhere, the Examiner resorts to "a matter of obvious design choice." But the obvious design choice reasoning is another way of saying "obvious to try," an approach soundly rejected by the Federal Circuit.

Claim 3 for example recites that the encoding logic generates an intermediate value by pre-pending a predetermined pattern of bits to the specified instruction address bits, from which the encoded instruction address can then be selected. The non-limiting example embodiment of Figure 6 illustrates such an approach (see elements 610, 620), which has been found to be particularly efficient for performing the defined encoding.

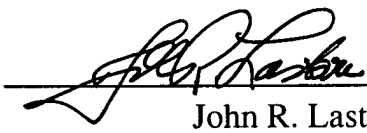
Claim 4 for example recites a form of compression logic that can be used to compress the encoded instruction address. The non-limiting example embodiment of Figure 5 illustrates such an approach (see also the text on page 16 of the application as originally filed). It has been found that such an approach produces a particularly efficient compressed encoding, which can result in a significant reduction in the bandwidth required to output such encoded instruction addresses.

VIII. CONCLUSION

Claim 21 is clear and distinct. The indefiniteness rejection is improper. The final prior art rejection fails to provide a prior art reference that teaches all the features of the claims. The Examiner tries to shore up the deficiencies in the rejection using an erroneous standard for obviousness and faulty analogies. The Board should reverse the outstanding rejections.

Respectfully submitted,

NIXON & VANDERHYE P.C.

By: 

John R. Lastova
Reg. No. 33,149

JRL/sd
Appendix A - Claims on Appeal

IX. CLAIMS APPENDIX

1. An apparatus for processing data, said apparatus comprising:

a processing circuit for executing processing instructions from any of a plurality of instruction sets of processing instructions, each processing instruction being specified by an instruction address identifying that processing instruction's location in memory, the instruction address having a predetermined number of bits irrespective of the instruction set to which the associated processing instruction belongs, but a different number of most significant instruction address bits needing to be specified in the instruction address to uniquely identify processing instructions in different instruction sets; and

encoding logic for encoding at least one instruction address with an indication of the instruction set corresponding to that instruction to generate an n-bit encoded instruction address, the encoding logic being arranged to perform the encoding by performing a computation equivalent to removing any least significant bits not forming the instruction address bits needing to be specified, and extending the specified instruction address bits to n-bits by prepending a pattern of bits to the specified instruction address bits, the number of least significant bits removed and the pattern of bits prepended being dependent on the instruction set corresponding to that instruction.

2. The apparatus as claimed in Claim 1, wherein for each instruction set a first pattern of bits prepended to the specified instruction address bits of an instruction address from that instruction set is related to a second pattern of bits prepended to the specified instruction address bits of instruction addresses of different instruction sets by shifting the first pattern of bits.

3. The apparatus as claimed in Claim 1, wherein the encoding logic is arranged to perform the encoding by performing a computation equivalent to generating an intermediate

value by pre-pending a predetermined pattern of bits to the specified instruction address bits of the instruction address and then selecting as the encoded instruction address n bits from the intermediate value.

4. The apparatus as claimed in Claim 1, further comprising compression logic for compressing a said encoded instruction address by performing a computation equivalent to partitioning that encoded instruction address into a plurality of x-bit sections, comparing each x-bit section with the corresponding x-bit section of a preceding encoded instruction address and outputting as a compressed encoded instruction address the most significant x-bit section that differs from the corresponding x-bit section of the preceding encoded instruction address, along with any less significant x-bit sections.

5. The apparatus as claimed in Claim 4, wherein the compression logic is arranged to associate with each x-bit section to be output from the compression logic a flag to indicate whether that x-bit section is the last x-bit section being output as the compressed encoded instruction address.

6. The apparatus as claimed in Claim 5, wherein if a plurality of x-bit sections are to be output from the compression logic, the plurality of x-bit sections are output sequentially starting with the least significant x-bit section.

7. The apparatus as claimed in Claim 5, wherein the compression logic is further arranged to expand to y bits each x-bit section to be output from the compression logic, with the most significant y-x bits containing the flag.

8. The apparatus as claimed in Claim 7, wherein the flag is a single bit.

9. The apparatus as claimed in Claim 8, wherein y is 8 and x is 7.

10. The apparatus as claimed in Claim 1, wherein the encoding logic comprises an n-bit selector logic unit for receiving the intermediate value and an identifier signal identifying the

instruction set associated with the instruction address contained within the intermediate value, the n-bit selector being arranged to output a predetermined n-bits of the intermediate value dependent on the identifier signal.

11. The apparatus as claimed in Claim 4, wherein the compression logic comprises a plurality of comparators, each comparator being arranged to receive a corresponding x-bit section of the encoded instruction address, and including temporary storage for storing the corresponding x-bit section of the preceding encoded instruction address, the comparator being arranged to compare the two x-bit sections and to generate a difference signal which is set when the two x-bit sections are different.

12. The apparatus as claimed in Claim 11, wherein the compression logic further comprises a flag generator logic arranged to generate for each x-bit section to be output from the compression logic a flag based on predetermined combinations of the difference signals generated by the plurality of comparators, such that a flag for a particular x-bit section is set if a more significant x-bit section is also to be output.

13. The apparatus as claimed in Claim 12, wherein the compression logic further comprises an output generator for generating the compressed encoded instruction address by pre-pending to each x-bit section to be output its corresponding flag, thereby generating as the output compressed encoded instruction address a sequence of y-bit sections.

14. The apparatus as claimed in Claim 4, wherein the encoding logic and compression logic are provided within a trace module used to trace activities of the processing circuit.

15. A tracing tool for a data processing apparatus, the data processing apparatus having a processing circuit for executing processing instructions from any of a plurality of instruction sets of processing instructions, each processing instruction being specified by an instruction address identifying that processing instruction's location in memory, the instruction address having a

predetermined number of bits irrespective of the instruction set to which the associated processing instruction belongs, but a different number of most significant instruction address bits needing to be specified in the instruction address to uniquely identify processing instructions in different instruction sets, and the tracing tool comprising:

encoding logic for encoding at least one instruction address with an indication of the instruction set corresponding to that instruction to generate an n-bit encoded instruction address, the encoding logic being arranged to perform the encoding by performing a computation equivalent to removing any least significant bits not forming the instruction address bits needing to be specified, and extending the specified instruction address bits to n-bits by prepending a pattern of bits to the specified instruction address bits, the number of least significant bits removed and the pattern of bits prepended being dependent on the instruction set corresponding to that instruction.

16. The tracing tool as claimed in Claim 15, further comprising compression logic for compressing an encoded instruction address by performing a computation equivalent to partitioning the encoded instruction address into a plurality of x-bit sections, comparing each x-bit section with the corresponding x-bit section of a preceding encoded instruction address and outputting as the compressed encoded instruction address the most significant x-bit section that differs from the corresponding x-bit section of the preceding encoded instruction address, along with any less significant x-bit sections.

17. A method of storing instruction set information, a processing circuit being arranged to execute processing instructions from any of a plurality of instruction sets of processing instructions, each processing instruction being specified by an instruction address identifying that processing instruction's location in memory, the instruction address having a predetermined number of bits irrespective of the instruction set to which the associated processing instruction

belongs, but a different number of most significant instruction address bits needing to be specified in the instruction address to uniquely identify processing instructions in different instruction sets, the method comprising the steps of:

encoding logic encoding at least one instruction address with an indication of the instruction set corresponding to that instruction to generate an n-bit encoded instruction address, by performing a computation equivalent to:

the encoding logic removing any least significant bits not forming the instruction address bits needing to be specified, and

the encoding logic extending the specified instruction address bits to n-bits by prepending a pattern of bits to the specified instruction address bits, the number of least significant bits removed and the pattern of bits prepended being dependent on the instruction set corresponding to that instruction.

18. The method as claimed in Claim 17, further comprising the step of compressing a said encoded instruction address by performing a computation equivalent to:

partitioning the encoded instruction address into a plurality of x-bit sections;

comparing each x-bit section with the corresponding x-bit section of a preceding encoded instruction address; and

outputting as the compressed encoded instruction address the most significant x-bit section that differs from the corresponding x-bit section of the preceding encoded instruction address, along with any less significant x-bit sections.

19. The method as claimed in Claim 18, further comprising decompressing the compressed encoded instruction address by performing a computation equivalent to:

determining the number of x-bit sections forming the compressed encoded instruction address; and

extending as necessary the compressed encoded instruction address to n-bits by incorporating additional x-bit sections obtained from corresponding x-bit sections of a preceding encoded instruction address, thereby producing the encoded instruction address.

20. The method as claimed in Claim 19, further comprising the step of decoding the encoded instruction address by performing a computation equivalent to determining from the predetermined pattern of bits the instruction set to which the instruction address relates, and removing the predetermined pattern of bits to yield the specified instruction address bits.

21. A computer readable medium carrying a computer program for controlling an apparatus in accordance with the method of claim 17.

22. The apparatus as claimed in claim 1, wherein the encoding logic is provided within a trace module used to trace activities of the processing circuit.

X. EVIDENCE APPENDIX

There is no evidence appendix.

XI. RELATED PROCEEDINGS APPENDIX

There is no related proceedings appendix.